Software

IDC DOCUMENTATION

WaveExpert

Approved for public release;
distribution unlimited

Notice
This document was published November 2000 by the Monitoring Systems Operation of Science Applications International Corporation (SAIC) as part of the International Data Centre (IDC) Documentation. Every effort was made to ensure that the information in this document was accurate at the time of publication. However, information is subject to change.

Contributors
Ethan Brown, Science Applications International Corporation

Ordering Information
The ordering number for this document is SAIC-00/3029.

This document is cited within other IDC documents as [IDC7.1.11].

WaveExpert

CONTENTS

WaveExpert

FIGURES

WaveExpert

TABLES

About this Document

This chapter describes the organization and content of the document and includes the following topics:

About this Document

PURPOSE

This document describes the design of the *WaveExpert* software of the International Data Centre (IDC). The software is a unit of the Network Processing CSC computer software component (CSC) of the Automatic Processing Computer Software Configuration Item (CSCI). This document provides a basis for implementing, supporting, and testing the software.

SCOPE

This document describes the architecture and detailed design of the software including its functionality, components, data structures, high-level interfaces, method of execution, and underlying hardware. This information is modeled on the Data Item Description for Software Design Descriptions [DOD94a] and Software Requirements Specification [DOD94b].

AUDIENCE

This document is intended for all engineering and management staff concerned with the design of all IDC software in general and of *WaveExpert* in particular. The detailed descriptions are intended for programmers who will be developing, testing, or maintaining *WaveExpert*.

RELATED INFORMATION

The following document complements this document:

- *IDC Processing of Seismic, Hydroacoustic, and Infrasonic Data* [IDC5.2.1]

See "References" on page 53 for a list of documents that supplement this document. The *WaveExpert* UNIX man page applies to the existing *WaveExpert* software.

USING THIS DOCUMENT

This document is part of the overall documentation architecture for the IDC. This document is part of the Software category, which describes the design of the software. This document is organized as follows:

- Chapter 1: Overview

  This chapter provides a high-level view of *WaveExpert*, including its functionality, components, background, status of development, and current operating environment.

- Chapter 2: Architectural Design

  This chapter describes the architectural design of *WaveExpert*, including its conceptual design, design decisions, functions, and interface design.

- Chapter 3: Detailed Design

  This chapter describes the detailed design of *WaveExpert* including its data flow, software units, and database design.

- References

  This section lists the sources cited in this document.

- Glossary

  This section defines the terms, abbreviations, and acronyms used in this document.

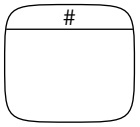
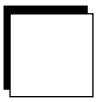
- Index

  This section lists topics and features provided in the document along with page numbers for reference.

Conventions

This document uses a variety of conventions, which are described in the following tables. Table I shows the conventions for data flow diagrams. Table II shows the conventions for entity-relationship diagrams. Table III lists typographical conventions. Table IV explains certain technical terms that are not part of the standard Glossary, which is located at the end of this document.

TABLE I: DATA FLOW SYMBOLS

Description[1]	Symbol
process	
external source or sink of data	
M = memory store T = tape store D = disk store Db = database store MS = mass store	
control flow	
data flow	

1. Symbols in this table are based on Gane-Sarson conventions [Gan79].

TABLE II: ENTITY-RELATIONSHIP SYMBOLS

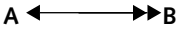
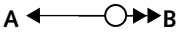
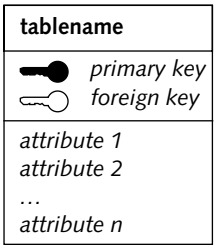
Description	Symbol
One **A** maps to one **B**.	
One **A** maps to zero or one **B.**	
One **A** maps to many **B**s.	
One **A** maps to zero or many **B**s.	
database table	

TABLE III: TYPOGRAPHICAL CONVENTIONS

Element	Font	Example
database table	**bold**	**dataready**
database table and attribute, when written in the dot notation		**prodtrack.***status*
database attributes	*italics*	*status*
processes, software units, and libraries		*ParseSubs*
user-defined arguments and variables used in parameter (par) files or program command lines		`delete-remarks` *object*
titles of documents		*Continuous Data Subsystem*
computer code and output	`courier`	`>(list 'a 'b 'c)`
filenames, directories, and websites		`ars.scm`
text that should be typed in exactly as shown		`edit-filter-dialog`

TABLE IV: TECHNICAL TERMS

Term	Description
CLIPS Expert System	An embedded system that applies rules to a set of facts and performs the actions specified. Rules are in the form of "`IF` <*condition1*>, <*condition2*>, ... `THEN` <*action1*>, <*action2*>, …" In the context of *WaveExpert*, rules can be written using the condition of facts about a station (for example, distance, importance) to cause an action of station rank value assignment.
fact	In the context of the *CLIPS* expert system, a fact is the fundamental unit of information used by the expert system rule set. Facts are expressed in the form of a list, for example (`station-rank ARCES 1.0`), or in the case of station-specific attributes provided by *WaveExpert*, a list of key/value pairs. For example, (`beta_station` (`sta ARCES`) (`azimuth 30.0`) (`delta 21.2`) …). See the *WaveExpert* man page for the complete list of parameters.
IADR	Interactive Analyst Data Request application, which allows an analyst to request auxiliary station data.
jackknife algorithm	Method in which one auxiliary station at a time is added to the set of event-associated stations to assess its potential for event-location improvement.
station rank	Estimation provided by *WaveExpert* of the improvement to an event location by including the specific station. The rank is defined as the decrease in the volume of the event error ellipsoid realized by including a station arrival in the location input. The rank can also be assigned to an arbitrary value by user-defined rules.
station importance	Value provided by the *libloc* routines indicating the importance of a particular station to the event solution. In general, stations filling a gap in a region where there are no other stations will have higher importance values.
ticron_server	Process that monitors the operations database and executes *WaveExpert* to process all origins within the specified time interval.

Chapter 1: Overview

This chapter provides a general overview of the *WaveExpert* software and includes the following topics:

Chapter 1: Overview

INTRODUCTION

The software of the IDC acquires time-series and radionuclide data from stations of the International Monitoring System (IMS) and other locations. These data are passed through a number of automatic and interactive analysis stages, which culminate in the estimation of location and in the origin time of events (earthquakes, volcanic eruptions, and so on) in the earth, including its oceans and atmosphere. The results of the analysis are distributed to States Parties and other users by various means. Approximately one million lines of developmental software are spread across six computer software configuration items (CSCIs) of the software architecture. One additional CSCI is devoted to run-time data of the software. This CSCI consists of the parameters for each software module, which are described in the man pages and software user manuals. Figure 1 shows the logical organization of the IDC software. The Automatic Processing CSCI automatically processes data to detect and measure signals and to define, locate, and characterize events. This CSCI consists of the following computer software components (CSCs):

- Station Processing

  This software scans data from individual time-series stations for characteristic changes in the waveforms (detection of onsets) and characterizes such onsets (feature extraction). The software then classifies the detections as arrivals in terms of phase type.

- Network Processing

  This software combines arrivals from several stations originating from one event and infers the location and time of its origin.

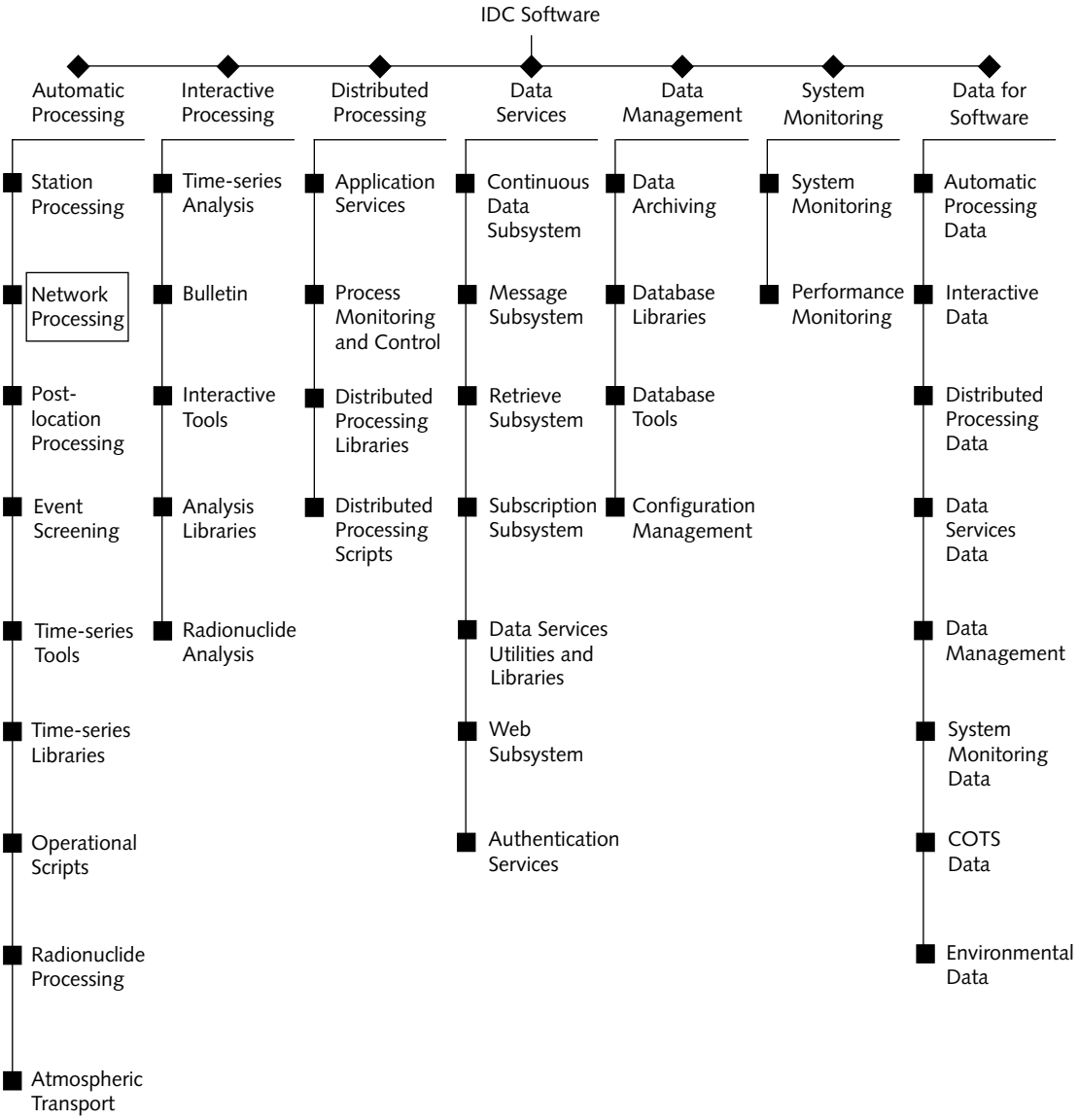


FIGURE 1. IDC SOFTWARE CONFIGURATION HIERARCHY

- Post-location Processing

  This software computes various magnitude estimates and selects data to be retrieved from auxiliary stations.

- Event Screening

  This software extracts a number of parameters that characterize an event; then a default subset of the calculated Event Characterization Parameters eliminates the events that are clearly not explosions.

- Time-series Tools

  This software includes various utilities for the Seismic, Hydroacoustic and Infrasonic (S/H/I) processing system.

- Time-series Libraries

  This software includes shared libraries to which several modules of the S/H/I processing system are linked.

- Operational Scripts

  This software provides miscellaneous functionality to enable automatic processing to function as a system.

- Radionuclide Processing

  This software includes the automated analysis, categorization, and flagging processes for radionuclide data.

- Atmospheric Transport

  This software includes the forward and backward modeling of the transport of particulates by atmospheric movements.

WaveExpert resides in the Automatic Processing CSCI and fulfills two roles in the IDC system. In its primary role, as part of the automatic processing pipeline, *WaveExpert* generates auxiliary station data requests based on primary network events. In addition, *WaveExpert* is part of the Interactive Analyst Data Request (IADR) system of the Interactive Processing CSCI.

Figure 2 shows the relationship of *WaveExpert*, in its automatic processing pipeline role, to the components of the Automatic Processing CSCI and selected components of the Data Services CSCI. The operations database is populated by events formed by the *Global Association (GA)* application, using data from the continuously transmitting stations of the primary S/H/I networks. Detections are available for event formation after signal detection and feature extraction and detection grouping and initial phase identification have been made by the *DFX* and *StaPro* applications. The *ticron_server* process periodically checks the operations database and creates new intervals for *WaveExpert* after *GA* has completed. The creation of the interval in the database is part of a transaction that includes insertion of a Tuxedo queue element/interval in a Tuxedo queue. If the Tuxedo queue has an interval, the interval queue element is forwarded to a *tuxshell* server, which in turn executes *WaveExpert*. *WaveExpert* computes time intervals for auxiliary network stations that potentially contain phase information to improve the existing event locations. The computed data intervals are written to the **request** table and then are used by the application *WaveGet_server* and subsequent processes to request and acquire the data. These data are then available to refine the location of the origin during subsequent reprocessing by *GA* or during interactive analysis.

Figure 3 shows the relationship of *WaveExpert*, in its IADR role, to selected components of the Interactive Processing and Data Services CSCIs. *WaveExpert* communicates with IADR using Interprocess Communication (IPC). If an analyst thinks additional auxiliary station data are required for event analysis, they initiate the IADR process. IADR sends a request to *WaveExpert* (ipc-assess mode), which in turn provides a list of station ranking assessments back to IADR. IADR graphically displays the assessments to the analysts and plots the stations on the Map. The analysts select the stations they feel will best constrain the event and IADR sends the information to *WaveExpert* (`ipc-request` mode). *WaveExpert* computes the intervals for the station list sent by IADR and writes them to the **request** table. The Data Services applications obtain the data and make them available to the analysts in *Analyst Review Station (ARS)*.

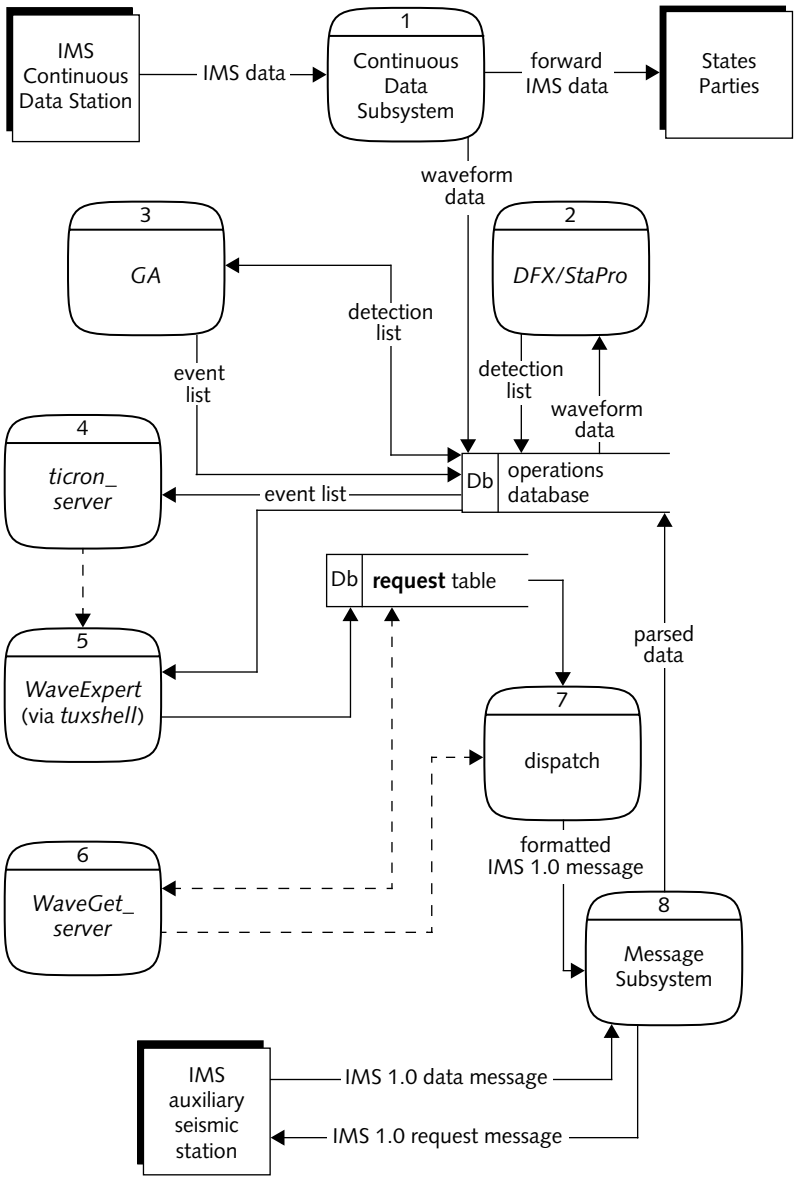


FIGURE 2. RELATIONSHIP OF WAVEEXPERT TO OTHER SOFTWARE UNITS OF AUTOMATIC PROCESSING CSCI

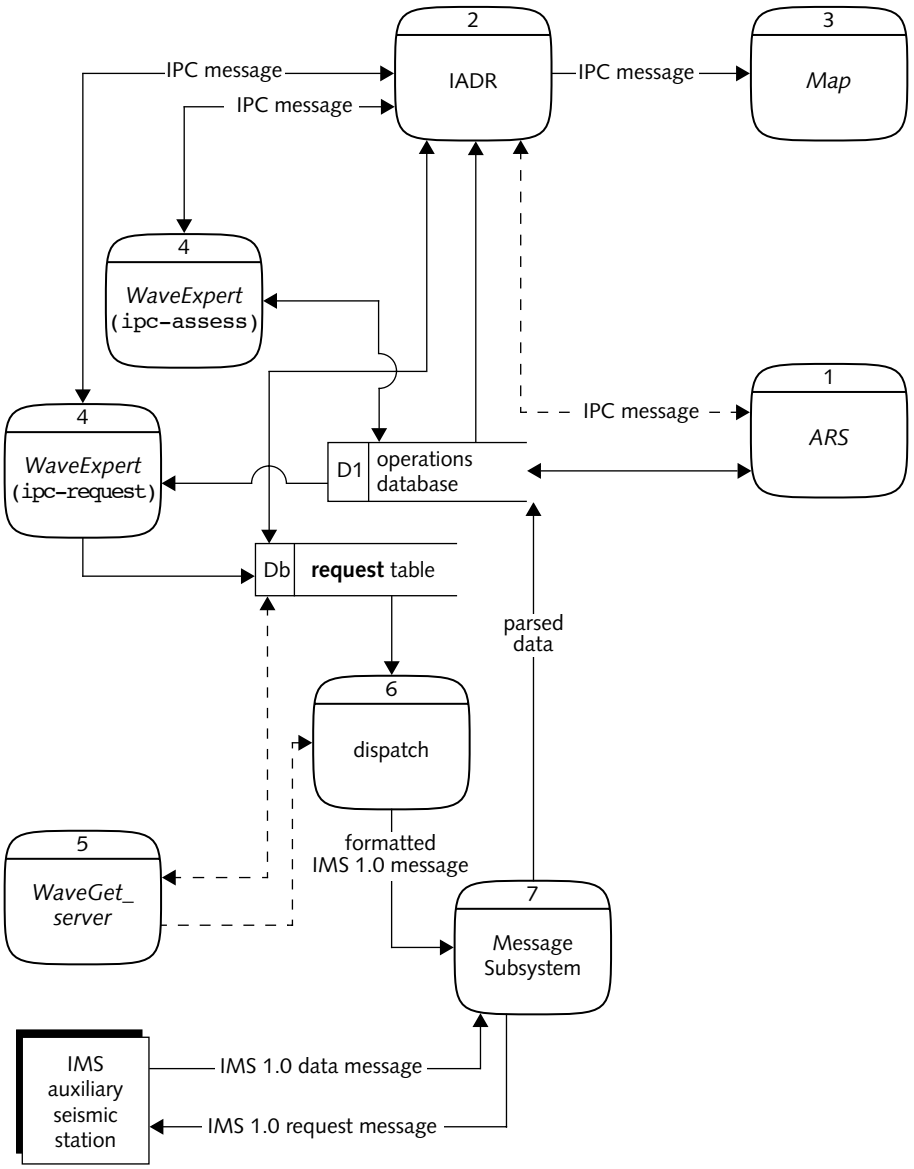


FIGURE 3. RELATIONSHIP OF WAVEEXPERT IADR CONFIGURATION TO SOFTWARE UNITS OF AUTOMATIC PROCESSING AND INTERACTIVE PROCESSING CSCIS

FUNCTIONALITY

WaveExpert, in its most basic functionality, generates data request intervals. When used with the Retrieve Subsystem, *WaveExpert* determines time intervals for stations in the auxiliary station set that may provide detections for improving origins formed using the primary station network arrivals. Additionally, as a part of the IADR system, *WaveExpert* first provides information to assist the analyst in selecting stations likely to improve an event's location and, after the analyst has made a station-set selection, computes and submits data intervals to the **request** table.

WaveExpert interacts directly with other software components only as part of the IADR, where requests and station rankings are passed via IPC. Otherwise *WaveExpert* is executed and receives input via a *tuxshell* process, initiated by ticron_server, and writes its results to the **request** database table.

IDENTIFICATION

WaveExpert is a self-contained component. Thus, the only component is identified as *WaveExpert*.

STATUS OF DEVELOPMENT

WaveExpert is a mature component of the IDC system and operated solely in an automatic auxiliary request mode for IDC Release 2. Release 3 also uses *WaveExpert* as part of the IADR system where it operates in concert with *ARS* and other interactive applications.

BACKGROUND AND HISTORY

Ethan Brown of Science Applications International Corporation (SAIC), Monitoring Systems Operation, developed *WaveExpert* in 1994. At that time the *Auxiliary* network was called the *Beta* network, and this nomenclature is used in function names and data structures throughout the *WaveExpert* source code. *WaveExpert* replaced the *wreq* application in the initial Beta Request System [Fox94].

WaveExpert, first used operationally in 1995, currently is an element of the PIDC at the CMR in Arlington, Virginia, U.S.A., and the International Data Centre of the Comprehensive Nuclear-Test-Ban Treaty Organization (CTBTO) in Vienna, Austria.

OPERATING ENVIRONMENT

The following paragraphs describe the hardware and commercial-off-the-shelf (COTS) software required to operate *WaveExpert*.

Hardware

WaveExpert is designed to run on a UNIX workstation such as the SPARC, manufactured by Sun Microsystems. Typically, the hardware is configured with at least 64 MB of memory and a minimum of 2 GB of magnetic disk space, although *WaveExpert* could be run on systems with less resources, depending on the number and type of other resident processes. The required disk space will scale with the quantity of logged messages that are archived; in normal operations with verbose output *WaveExpert* generates about four megabytes of output per day. *WaveExpert* must obtain other services (such as database access) over a Local Area Network (LAN) connection to other computers. Figure 4 shows a representative hardware configuration.

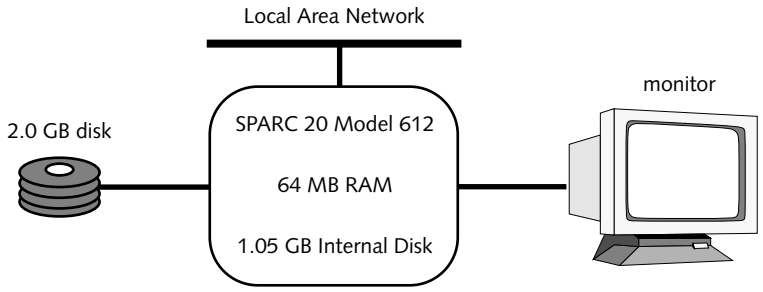


FIGURE 4. REPRESENTATIVE HARDWARE CONFIGURATION FOR WAVEEXPERT

Commercial-Off-The-Shelf Software

WaveExpert is released under the Solaris 7 operating system and ORACLE 8.1.5.

Chapter 2: Architectural Design

This chapter describes the architectural design of *WaveExpert* and includes the following topics:

Chapter 2: Architectural Design

CONCEPTUAL DESIGN

WaveExpert's purpose is to generate station-specific data request intervals for events. These intervals are used by other processes in the Retrieve Subsystem (see Figure 2 on page 6 and Figure 3 on page 7) to obtain data from remote sites. It was designed not only to be simple to operate using its default configuration, but also to provide a high degree of flexibility to satisfy general processing or station-specific needs. Figure 5 shows the processing flow for *WaveExpert*. *WaveExpert* has various modes of operation as specified by *mode* parameter values: `auto`, `ipc-assess`, `ipc-request`, and `interval`. As is the case with nearly all software applications, *WaveExpert* receives input, processes the input, and generates output. The nature of the mode-specific input, processing, and output are as follows:

auto Mode

This mode is used by *WaveExpert* as part of Automatic Processing to generate station data interval requests.

- Initiation:
  - invoked by *tuxshell*, periodically initiated by *ticron_server* to run following *GA* processing
- Command-line Input:
  - origin or time interval containing origins
  - network or station list
- Processing:
  - station selection by default algorithm and user-specified rules
  - interval computation based on origin time and phase travel times

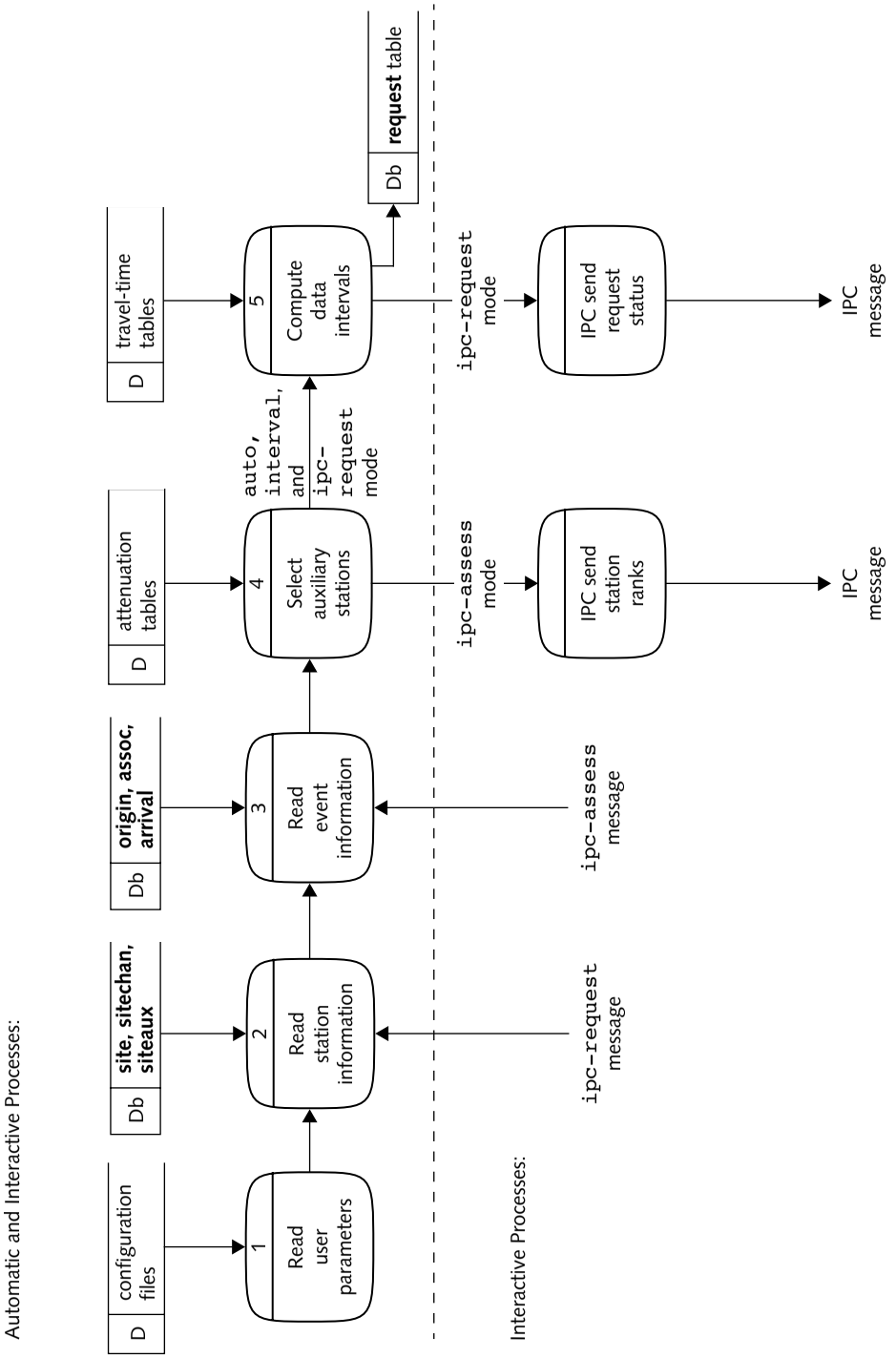


FIGURE 5. PROCESSING FLOW AND CREATION OF DATA PRODUCTS

- Output:
  - station intervals written to the **request** database table

ipc-assess Mode

This mode, as part of the IADR system, provides an analyst with an assessment of station potentials for refining an event location.

- Initiation:
  - IPC message from IADR
- Command-line Input:
  - origin and station network
- Processing:
  - detection probability and station rank computation
- Output:
  - station probability of detection, rank, event-station distance, and azimuth values are provided to IADR by an IPC message

ipc-request Mode

This mode generates station data **request** database table entries as part of the IADR system.

- Initiation:
  - IPC message from IADR
- Command-line Input:
  - origin and station list
- Processing:
  - interval computation for stations listed in command-line input, based on origin time and phase travel times

- Output:
  - data intervals for the input stations written to the **request** database table

interval Mode

This mode is used for long-period data interval requests.

- Command-line Input:
  - origin or time interval containing origin set
  - network or station list
  - request interval

  Processing:
  - compute request intervals based on network or station list and request interval
- Output:
  - station intervals written to the **request** database table

Input

Input to *WaveExpert* is obtained from user parameters, database tables and IPC messages. User parameters provide information such as the time interval containing origins to process, a specific origin to process, input and output database accounts, and values used for processing, such as threshold values (see the *WaveExpert.1* UNIX man page for a description of all parameters). IPC message input is provided in parameter (*name* = *value*) form. Static configuration parameters are provided by par files residing in the configuration area. Dynamic parameters particular to a specific execution are provided either on the command line by *tuxshell* or in the IPC messages string when *WaveExpert* is run as part of IADR. The database provides origin- and station-related information used for station selection and interval computation.

Processing

WaveExpert processing can provide a station set, the time interval to request for given stations, or both, depending on the processing mode in which it is executed. For selection of a station set, stations from the auxiliary network are ranked first by their probability of detecting the event and then by how much they will improve the event location. Other criteria may be used for ranking stations through use of the CLIPS embedded rule-based expert system [Ril98]. Station time interval requests are computed using the event origin time, station distance, and the theoretical travel times of the specified (via parameter) phase set.

Output

WaveExpert writes the request time intervals to the **request** database table (with the exception of the `ipc-assess` mode). Informational messages are written to the standard output and optionally to a log file (Table 1).

TABLE 1: STANDARD PRODUCTS AVAILABLE THROUGH WAVEEXPERT

Product	Availability Time (GMT[1])	Tailoring Criteria
Request Tuples	~1 minute	station, channel, origin ID
Station Ranking (`ipc-assess` mode)	~3 seconds	origin ID

1. Greenwich Mean Time

WaveExpert's general processing flow is shown in Figure 5. The flow is somewhat mode specific, but the simplified flow is as follows:

(1) *WaveExpert* reads command line and configuration files, which provide the station list and database table names to the next process. (2) *WaveExpert* reads the database table **site**, **sitechan**, and **siteaux** and the station list, which during `ipc-request` mode is provided by an IADR IPC message. (3) *WaveExpert* then reads event information from the **origin**, **assoc**, and **arrival** tables, using parameters to provide the *orid* and database table names. During `ipc-assess` mode an IADR

IPC message provides the *orid*. (4) *WaveExpert* selects auxiliary stations. If the processing mode is `ipc-assess`, an IPC message is sent back to IADR, and processing for this *orid* finishes. (5) *WaveExpert* computes the data intervals and sends output to the **request** database table. An IPC message is returned to IADR during `ipc-request` mode.

DESIGN DECISIONS

The following design decisions pertain to *WaveExpert*.

Programming Language

Each processing unit of *WaveExpert* is written in the C programming language. Users may provide rules for auxiliary station selection via the *CLIPS* rule-based expert system syntax.

Global Libraries

The software of *WaveExpert* is linked to the following libraries: *libaesir, libgd, libgeog, libinterp, libLP, libloc, libpar, libprob,* and *libstdtime*.

The software is also linked to the following COTS library: *libclips.*

Database

WaveExpert uses tables of the *Database Schema* [IDC5.1.1Rev2] for input and output.

- Input: **site, sitechan, siteaux, origin, origerr, assoc, arrival, affiliation,** and **wfdisc**
- Output: **request**

Interprocess Communication

WaveExpert, as a component of IADR, uses IPC according to its operational mode:

ipc-assess Mode

- *WaveExpert* receives a string of *name = value* pairs that provide the origin to process, database accounts, and non-standard (*ARS* temporary) table names.
- *WaveExpert* returns a list of stations with the station detection probabilities, ranks, distances, and azimuths in the following form:

  "sta1:prob1:rank1:dist1:az1,sta2:prob2:rank2:dist2:az2,..."

ipc-request Mode

- *WaveExpert* receives a string of *name = value* pairs that provide the origin to process, the station list for which to request intervals, database accounts, and non-standard (*ARS* temporary) table names.
- *WaveExpert* returns a list of stations with intervals in the following form:

  "sta1:start1:end1:,sta2:start2:end2,..."

File System

WaveExpert uses the file system in various ways including to customize processing through parameter settings, to specify earth models, and to log program output. These data are stored in the configuration and log directories.

- Parameter files are used to configure input and output and to customize processing.
- Rule files are used to define specialized station ranking rules.
- Travel-time files are used to compute request intervals based on phase travel times and in *libloc* functions as part of the station ranking process. The travel-time files are referenced through a velocity model specification file provided by the parameter *vmsf.*
- An attenuation file is used to compute station detection-probability and is specified by the parameter *atten_table*.
- *WaveExpert* writes its informational messages to log files.

Design Model

The design of *WaveExpert* was primarily influenced by requirements pertaining to the cost of communications to auxiliary stations, timeliness, flexibility, and reliability. Flexibility is provided by a rich set of available user parameters and by the embedded *CLIPS* Expert System, which allows arbitrary station selection rules to be configured by the user without rebuilding/releasing *WaveExpert* sources or executables.

Database Schema Overview

WaveExpert uses the ORACLE database for the following purposes:

- to obtain station information such as location and noise characteristics
- to obtain event information such as location, magnitude, and associations
- to obtain information on existing or previously requested data intervals
- to store its final request interval product

Table 2 shows the database tables used by *WaveExpert*. The first column identifies the database table. The Mode field is "R" if *WaveExpert* reads from the table and "W" if the system writes to the table.

TABLE 2: DATABASE TABLES USED BY WAVEEXPERT

Name	Mode	Description
affiliation	R	groups stations into networks and elements into arrays
arrival	R	contains data on a signal that has been identified as an arrival
assoc	R	contains data that relate arrivals to origins
origerr	R	contains uncertainties/errors in the origins
origin	R	describes an origin (location, magnitude, and so on)
request	R/W	contains request time intervals

TABLE 2: DATABASE TABLES USED BY WAVEEXPERT (CONTINUED)

Name	Mode	Description
site	R	contains data on a station site, such as latitude, longitude, and elevation
siteaux	R	contains ancillary station information such as ambient noise levels
sitechan	R	contains channel specific information for a site, such as channel orientation and sample rate
wfdisc	R	contains information pertaining to time-series data (S/H/I)

FUNCTIONAL DESCRIPTION

WaveExpert's processing functions are shown in Figure 6 and are described as follows:

- Initialization

  Read processing parameters from the command line and configuration files. Run the start hook.

- Get Site Information

  Load the site-related information from the database into *WaveExpert*'s internal data structures.

- Get Origin Information

  Load the origin-related information from the database into *WaveExpert*'s internal data structures.

- Compute Station Detection Probability

  Use the site, origin, attenuation information, and parameter information to compute the detection probability at each station for the event being processed.

- Default Station Selection (Jackknife)

  Rank each station in terms of its potential for improving the event location. Select the final station set for data requests.

- Station Selection (rule based)

  Rank each station according to an optional user-specified rule set.

- Station Interval Computation

  Use the site, origin, and travel-time information to compute the data intervals to request.

- Interval Output

  Store the intervals computed in Station Selection (rule based) in the database for use by other applications in the Retrieve Subsystem; the final goal is to transport the **request** table data intervals from the remote station data store to the local analysis system.

Initialization

Initialization is represented by process 1 in Figure 6. The top-level function (main) calls the parameter input functions, an optional start-hook, the mode-specific processing function, and then the end-hook. The start- and end-hooks provide the system with a way to run an arbitrary command–commonly a "print-date" type of script–supplied by a parameter.

Parameter Input

Parameter input is also represented by process 1 in Figure 6. This process verifies that parameters required for processing are provided, sets default values for optional parameters, and loads the parameter values into program memory. Most actions are achieved through the *libpar* functions; *libpar* is the standard parameter interface library for the IDC system.

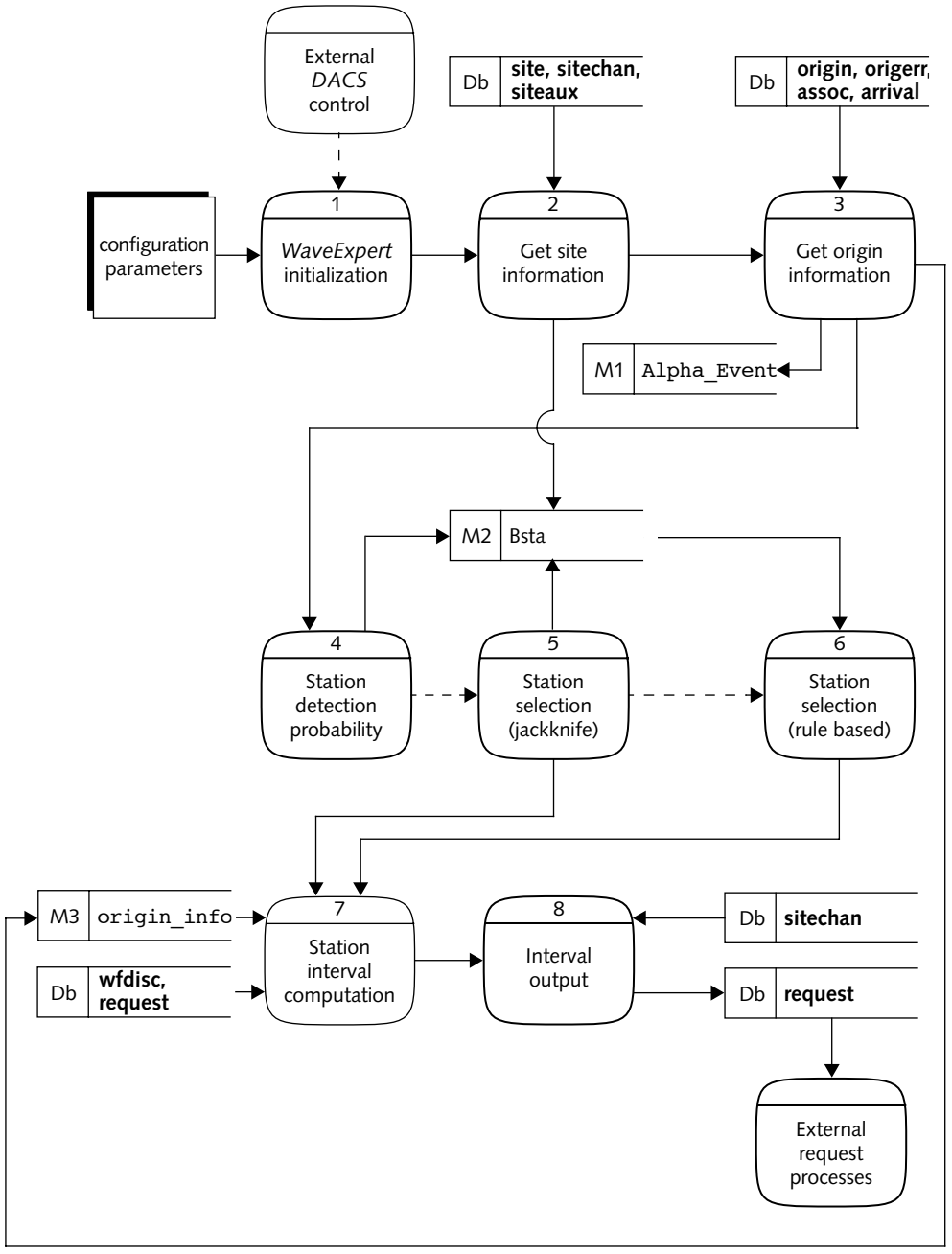


FIGURE 6. FUNCTIONAL DESIGN OF WAVEEXPERT

Database Site Input

Database Site Input is represented as process 2 in Figure 6. Site-related information for the auxiliary stations is loaded into an array of Bsta structures (see Table 3 on page 31) for convenient access during processing. If a network is specified, then information for all stations contained in the network, as defined in the **affiliation** table, are loaded. Otherwise, only data for stations in the given station list are loaded. All site information is obtained from the database using *libgdi* functions.

Database Origin Input

Process 3 in Figure 6 reads origin-related information, which includes **origin**, **origerr**, **assoc**, and **arrival** records, and site information for the associated primary stations from the database. It stores this information in the `Alpha_Event` structure (see Table 5 on page 48) for convenient access during program processing. All origin information is obtained from the database using *libgdi* functions

Station Detection Probability

Process 4 in Figure 6 computes the probability that an arrival will be detected at a specific station for the given origin. Detection probability is used to limit the station set analyzed by the station selection process to stations likely to detect an event and to order stations during the ranking process. The detection probability calculation uses the station distance, event magnitude, amplitude attenuation, and the station's nominal ambient noise level to arrive at the probability estimation.

Default Station Ranking

Process 5 in Figure 6 ranks stations from the decrease in the event error ellipsoid assuming an auxiliary station arrival is available to the event location computation. A jackknife algorithm, which adds the auxiliary stations to the original set of associated primary stations by order of detection probability is used. The rank value assigned is the decrease in the location error ellipsoid volume when a theoretical

arrival from the auxiliary station is included in the location. If the error improvement exceeds a threshold, the auxiliary station is included in the association set used for following station rankings, and the next auxiliary station is processed.

Rule-based Station Ranking

The default ranking value provided by Default Station Ranking can be modified or overridden through user-specified rules (process 6 in Figure 6). Information about the station can be used via an internal *CLIPS* rule-based expert system [Ril98] to provide extremely flexible ranking schemes without the need to modify the existing *WaveExpert* source code. In the current *WaveExpert* configuration, a rule exists to force all stations within five degrees to be requested. The following example shows a simple rule to assign the station rank to be the detection probability:

```
Example: (defrule we-simple
         ?f1 <- (beta_station (sta ?s) (det_prob ?p))
         =>
         (assert (station-rank ?s ?p)))
```

Rules operate on facts, which are instantiated by *WaveExpert* in the form of a beta_station template containing the attributes *sta, azimuth, delta, import, solo_import, gap_redux, jacknife, det_prob,* and *err_vol_change*. For more information on writing rules see the *WaveExpert* UNIX man page.

Station Interval Computation

WaveExpert computes a request interval (process 7 in Figure 6) to encompass all phases of interest (as specified by the parameter *phases*). Theoretical travel times for all phases are computed via functions in the *libloc* library, and the earliest and latest phase arrival times are used as the basis for the interval. No extrapolation is used, so only distance-appropriate phases are included. In practice, the phase list is commonly P, Pn, Pg, Lg, and Rg, so regional-distance stations render larger intervals (to include Pn through Lg or Rg) than teleseismic-distance stations that include only the P phase.

After the intervals have been computed, existing intervals–either previously requested or already resident in the **wfdisc** table–are accounted for and removed from the computed interval set.

Interval Output

Interval output, process 8 in Figure 6, determines the station-related channel set written to the **request** table. The channel set is selected based on the parameter *expand-array*, which causes individual array elements to be requested explicitly, rather than a single entry for the array name. Channel components are selected based on the general parameter *components* or the station-specific parameter *<STA>-components* and the element's entries in the **sitechan** table.

INTERFACE DESIGN

This section describes *WaveExpert*'s interfaces with other IDC systems, external users, and operators.

Interface with Other IDC Systems

In the Automatic Processing pipeline, the *WaveExpert* process sits at the junction of automatic event processing (*GA*) and the Retrieve Subsystem. At the time that *ticron_server* executes *WaveExpert*, event formation has completed, and the pipeline is ready to obtain auxiliary station data to improve the events formed from the primary station arrivals.

WaveExpert receives its processing configuration through a *tuxshell*, responding to a message from the *ticron_server* process. *WaveExpert* obtains origin- and site-related database information and writes its results through *libgdi* function calls.

During interactive processing, *WaveExpert* communicates with elements of the IADR system through IPC messages. *WaveExpert* responds to the `ProcessOrid` message ID and parses the processing parameters out of the IPC message string. *WaveExpert*, in `ipc-assess` mode, responds with a message ID of `WEAssessSuccess` and a message containing station ranking information or

`WEAssessFail` if a problem occurs during processing. In `ipc-request` mode a status message ID of `WERequestSuccess` or `WERequestFail` is returned to the calling process with a message string containing station request intervals.

Interface with External Users

WaveExpert has no interface with external users.

Interface with Operators

WaveExpert provides informational messages to standard output and to an optional log file, when the *log_file* parameter is set, which can provide clues to the operator should processing fail. The informational output may also help in general tuning of the application. The *Workflow* application (when configured to monitor the request table) is useful in tracking the operational behavior of *WaveExpert*.

Chapter 3: Detailed Design

This chapter describes the detailed design of *WaveExpert* and includes the following topics:

Chapter 3: Detailed Design

DATA FLOW MODEL

WaveExpert reads parameters and database information, computes station detection probabilities and ranks, and writes data intervals to the **request** database table. Figure 7 shows the data flow when *WaveExpert* is used to compute station intervals without selecting stations. Figure 8 shows the data flow when *WaveExpert* is used to select stations and compute station intervals. Flows in each figure split to describe the differences during ipc-type (IADR) mode. Because of the variation in processing between modes, some processes are not shown in both figures. External data stores (for example, the database) are indicated by D*n* objects. Program memory data stores are indicated by M*n* objects.

PROCESSING UNITS

The actions these processing steps represent are achieved through the *WaveExpert* functions shown in Figures 7 and 8 and Table 3. The following paragraphs describe the design of these processing steps in relation to the *WaveExpert* functions, including any constraints or unusual features in the design. The logic of the software and any applicable procedural commands are also provided.

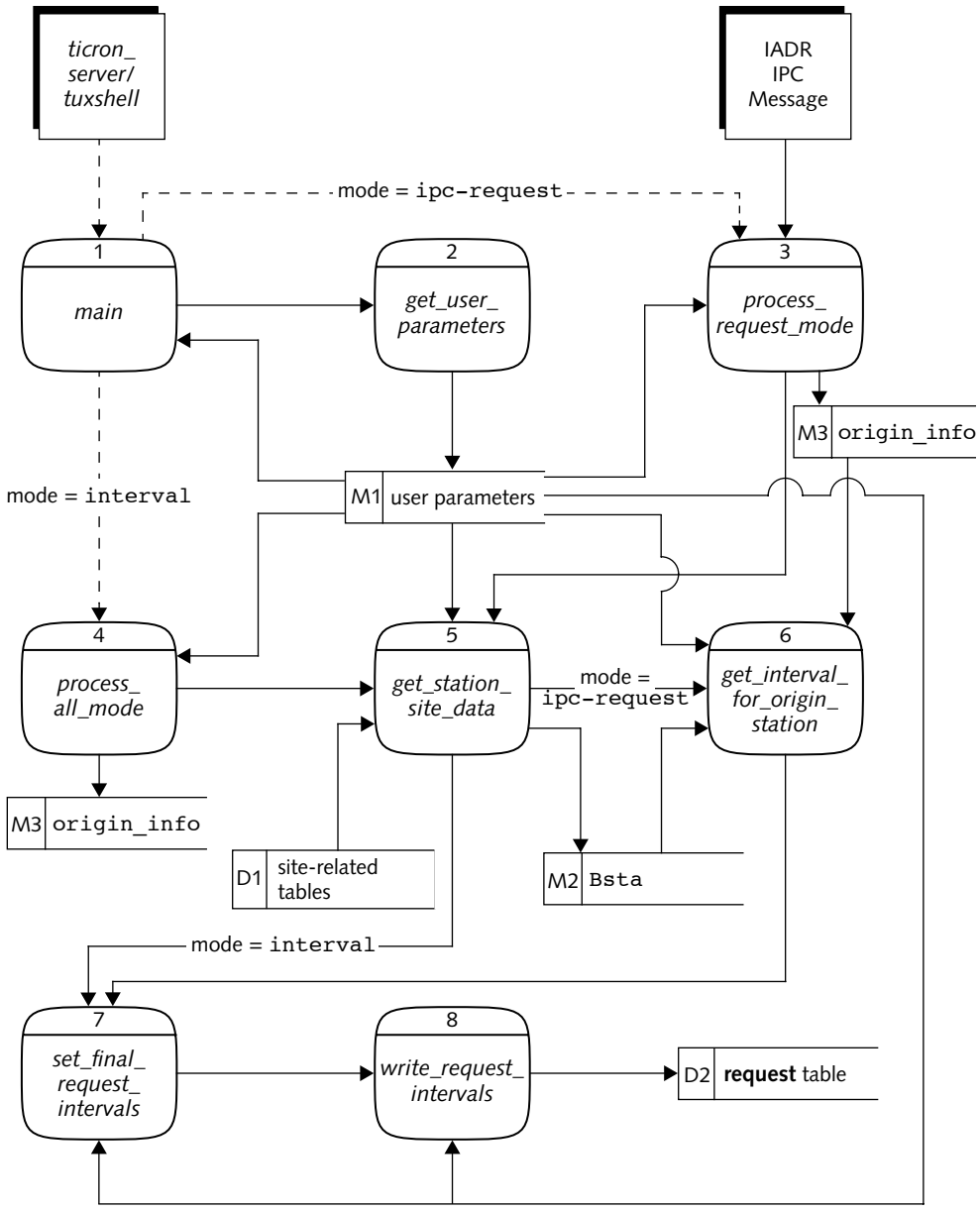


FIGURE 7. DATA FLOW MODEL OF WAVEEXPERT, STATION INTERVAL COMPUTATION ONLY

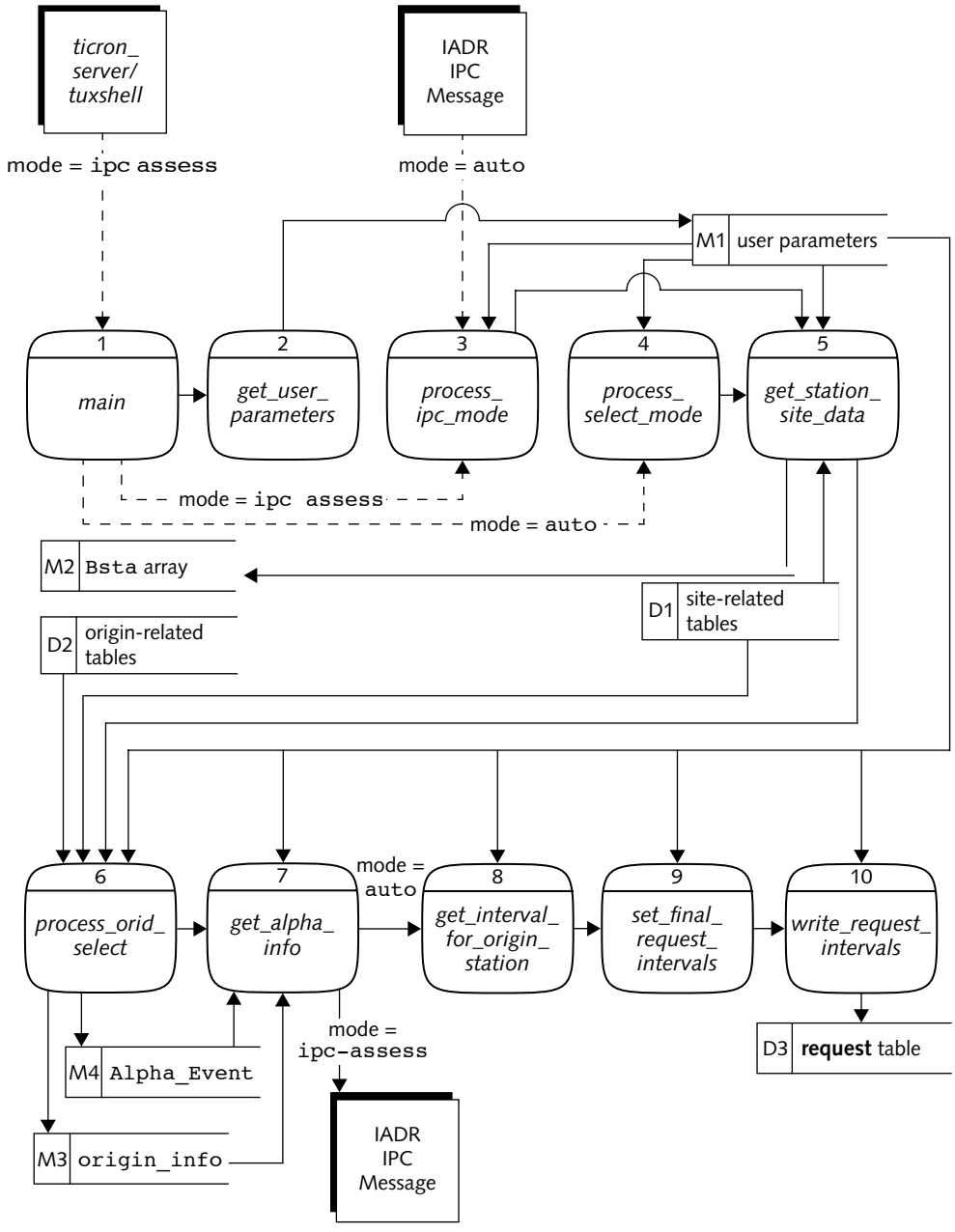


FIGURE 8. DATA FLOW MODEL OF WAVEEXPERT STATION SELECTION AND STATION INTERVAL COMPUTATION

TABLE 3: RELATIONSHIP BETWEEN PROCESSING STEPS AND FUNCTIONS

Processing Step	Functions
Main	*main*
Parameter Input	*get_user_parameters*
Processing Mode	*process_request_mode, process_all_mode, process_ipc_mode, process_select_mode*
Database Input	*get_station_site_data, get_alpha_info*
Station Detection Probability	*process_orid_select*
Default Station Ranking	*process_orid_select*
Rule-based Station Ranking	*process_orid_select*
Station Interval Computation	*get_interval_for_origin_station, set_final_request_intervals*
Interval Output	*write_request_intervals*

Main

Main is the top-level *WaveExpert* function (process 1 in Figures 7 and 8), which initializes the *libgdi* database access library, user parameters, and log file, and calls start- and end-hooks. Start- and end-hooks are user-specified programs that execute prior to and following *WaveExpert* execution. The mode-specific processing path is defined at this point.

Input/Processing/Output

Main obtains its input from command-line arguments provided by the UNIX operating system, specifically the number of parameters on the command line and the array of parameters. Additional input is obtained from configuration parameters set during parameter initialization (see Parameter Input).

Processing is described in the introductory paragraph.

The output of this process is the exit status.

Control

Main is started by a *tuxshell* under control of the *ticron_server*. *Main* terminates upon completion of its subfunctions.

Interfaces

The *WaveExpert main* function interfaces through its calling arguments on the command line, user parameters (which are typically set to global variables accessible to other functions), arguments provided to subfunctions, and the log file.

Error States

The *WaveExpert main* function checks for error status return values after calls to its subfunctions. *WaveExpert* has been coded defensively; all functions return values for error status. Appropriate error messages indicating the failing subfunction are provided to standard output and the log file. Subfunction errors cause an exit status of ERR defined as `-1` in `aesir.h`.

Parameter Input

Parameter Input, process 2 in Figures 7 and 8, parses user parameters using the *libpar* routines. The parameters are checked to verify that required parameters (such as the input database) are provided. Default values are assigned to the optional parameters if they are not explicitly set.

Input/Processing/Output

This process obtains input from arguments that provide the number of parameters and the array of parameters in the form of the *argc argv* command line information passed to the *main* routine.

Processing within the Parameter Input process consists of setting the values of global parameter variables to the values specified by the parameter or to default values. Required parameter inputs are verified, or an error status is returned to the calling function.

Output of this process is achieved through setting values obtained through configuration parameters to global variables defined in `global_defs.h`.

Control

The Parameter Input function, *get_user_parameters*, is called from the *main* function. Control is returned to *main* after parameters have been processed.

Interfaces

During Parameter Input processing the global variables declared and made available through the include files, `global_defs.h` and `user_params.h`, are set to the parameter values via *libpar* function calls.

Error States

The most common errors that will occur in Parameter Input processing will be related to missing required parameters, such as databases or processing intervals. Required parameters can be identified in the *WaveExpert* man page as parameters having no default value. Should this type of error occur an appropriate message is printed, and an error status is returned to the calling function.

Processing Mode

The Processing Mode step controls the processing flow and calls *WaveExpert* functions appropriate to the processing mode specified by the *mode* parameter. In the IPC modes, for example, the message loops take place during this process.

Input/Processing/Output

The Processing Mode functions receive input from user parameters obtained in the Parameter Input process. No formal parameters are provided by the parent function *main*.

Processing within the Processing Mode step is specific to the function being executed. The functions *process_request_mode, process_all_mode, process_ipc_mode,* and *process_select_mode* are called from the *main* function according to the value of the *mode* parameter.

auto Mode

The *main* function calls *process_select_mode,* which connects to the input database, sets up the auxiliary station list, initializes and populates the `Bsta` structure array, obtains the origins to process, and then loops over each origin executing the main processing functions within a call to *process_orid_select.*

ipc-assess mode

The *main* function calls *process_ipc_mode,* which connects to the input database, initializes the IPC session, sets up the auxiliary station list, initializes and populates the `Bsta` structure array and disconnects from the database. All of the initialization processing is done before the IPC message loop is entered so *WaveExpert* can provide a rapid response time during interactive processing. Next, the IPC loop is entered, and *WaveExpert* listens for a message from IADR. When a message is received, the message arguments are parsed in the same manner as the user parameters (because IPC message arguments are in the par *name* = *value* form). The message specifies the database being used for the interactive session and the origin to be assessed. *WaveExpert* connects to the interactive database and calls *process_orid_select,* which makes the station assessments and sends the IPC message when run in `ipc-assess` mode. The function *process_ipc_mode* then goes to the top of the IPC loop and listens for the next IPC message. The IPC loop never exits, so *WaveExpert* must be terminated via signal when run in this mode.

ipc-request Mode

The *main* function calls *process_request_mode,* which initializes the IPC session and enters the IPC loop. *WaveExpert* listens for a message from IADR. When a message is received, the message arguments are parsed in the same manner as the user

parameters (because IPC message arguments are in the par *name* = *value* form). The message provides the database being used for the interactive session, the station list to be requested, and the origin to be assessed. The `Bsta` structures are populated for the message-specified station list, the `Origin_Info` structure is populated, and the intervals are computed for each station using the function *get_interval_for_origin_station.* These preliminary intervals are stored in the return IPC message string, because they may be modified in later processing due to existing data. The input database is closed, and the output database is opened. Next, the final intervals are set in the function *set_final_request_intervals* (taking into account existing data) and written to the database by the function *write_request_intervals.* The return IPC message is sent back to IADR, and the function *process_request_mode* returns to the top of the IPC loop and listens for the next IPC message. The IPC loop never exits, so *WaveExpert* must be terminated via signal when run in this mode.

interval Mode

The *main* function calls *process_all_mode,* which connects to the input database, sets up the auxiliary station list, initializes, and populates the `Bsta` structure array. The `Origin_Info` structure is populated, and the intervals are computed for each station using the function *get_interval_for_origin_station.* The input database is closed, and the output database is opened. Next, the final intervals are set in the function *set_final_request_intervals* (taking into account existing data), and written to the database in the function *write_ request_intervals.*

The IPC functions associated with this process provide output in the form of IPC messages to IADR. The non-IPC functions do not provide output.

Control

The Processing Mode functions are called from the *main* function. Control is returned to *main* only during non-IPC processing. Control does not return from the IPC Processing Mode functions after the IPC message handling loops have been entered.

Interfaces

During the Processing Mode step the data are exchanged through function calls and variable assignment. Processing controls are achieved through function return values and global variables.

Error States

The most common errors that will occur in the Processing Mode step will be related to database connection problems or errors states occurring in subfunctions returned to the Processing Mode functions. Non-IPC functions print an informational message and pass the error status to *main*, which in turn exits with an error status. When IPC functions encounter errors before entering the IPC message loops, they also pass an error status back to *main*. Errors occurring inside the IPC message loop cause an error status to be returned to IADR via an IPC message, and process control returns to the top of the IPC message loop.

Database Input

The Database Input process, represented by the functions *get_station_site_data* (process 5 in Figures 7 and 8) and *get_alpha_info* (process 6 in Figure 8), reads **origin** and **site**-related information from the database, which is used for the ranking and time interval computations. The function *get_station_site_data* obtains the station locations and other **site**- and **siteaux**-related attributes for a set of stations or network (usually the auxiliary network) and fills in the `Bsta` structure members (see Table 3), represented by object M2 in Figures 7 and 8. The function *get_alpha_info* (process 6 in Figure 8) obtains **origin**- and **site**-related information for associated stations and fills in the `Alpha_Event` and `Origin_Info` structures (see Tables 5 and 6), shown as objects M3 and M4 in Figure 8. Associated station information is used only during auxiliary station selection, so this process is not shown in Figure 7.

Input/Processing/Output

Input for *get_station_site_data* is an array of station informational structures, (object M2 in Figure 7 the `Bsta` structure, Table 3) having, at this point in the processing, only the station names initialized.

The processing entails setting up a list of sites and then using the list to query the **site**, **sitechan**, and **siteaux** tables. The information obtained from the queries populates (object M2 in Figure 7) members of the `Bsta` structure array argument for output.

Processing for the origin information is similar. The origin ID specified via parameter and an `Alpha_Event` structure (see Table 5) are passed in, then the **origin**, **origerr**, **assoc**, and **arrival** tables are queried. Stations obtained from the **assoc** query are then used to obtain the site information for the event-associated stations. Query results populate the `Alpha_Event` structure (object M3) for output.

Control

Database Input functions are called from the mode-specific processing routines: *process_select_mode* for *mode* = `auto`, *process_all_mode* for *mode* = `interval`, *process_request_mode* for *mode* = `ipc-request`, and *process_ipc_mode* for *mode* = `ipc-request`. The Database Input functions terminate by passing a status code to the calling functions.

Interfaces

The data used for processing in the Database Input functions are predominantly obtained from the Database Management System (DBMS) through calls to the *libgdi* library functions. The data are then stored in the `Bsta`, `Alpha_Event`, and `Origin_Info` data structures as shown in Figures 7 and 8, and described in Tables 4, 5, and 6, respectively.

Error States

Errors occurring in the Database Input functions will most likely be from database-related problems caused by missing tables, erroneous table names, or data inconsistencies (for example, associated stations having no site record). If an error occurs in the Database Input functions, an appropriate message is output, and the status is returned to the calling function, which in turn prints an appropriate message and passes an error status to its parent.

Station Detection Probability

Station Detection Probability is completed through the subfunction *process_orid_select* (process 7 in Figure 8), which is called from the *process_select_mode* and *process_ipc_mode* functions (processes 3 and 4 in Figure 8). The process uses the functions *get_det_prob* and *get_regional_prob* (described in the next section) to compute an estimate of the probability that a signal from an event will be detected at a specific station. This information is used to order the stations for the Default Station Ranking process and is available for the Rule-based Station Ranking process.

Input/Processing/Output

The Station Detection Probability process consists of subfunctions to *process_orid_select* (process 7 in Figure 8): *get_det_prob* for general probability and *get_regional_prob* for regional-distance-specific probability. The regional probability computation takes into account local attenuation characteristics and is preferable to the general probability provided by *get_det_prob* when regional attenuation information is available.

The input arguments to *get_det_prob* are an array of `Bsta` structures for all stations being processed and an `Origin` structure (a member of the `Alpha_Event` structure, which is a "C" structure representation of an **origin** table) containing information pertaining to the event being processed. Other non-argument input is comprised of detection thresholds provided by user parameters. The actual proba-

bility computation is performed by the *libprob* function *prob_of_detection*, and the resulting detection probabilities are stored in the `det_prob` member of the station's `Bsta` structure.

Input arguments for *get_regional_prob* are an `Origin` structure and a single station's `Bsta` structure. Like the value from *get_det_prob*, the detection probability is stored in the `det_prob` member of the station's `Bsta` structure. The regional value, when available, will supersede the general probability value computed *get_det_prob.* The regional probability is estimated as the normalized probability function value *phi(x),* where

$$y = \frac{\log(Amp) - \log(Noise) - \log(snr - thresh)}{\sqrt{\sigma_{amp}^{\ 2} + \sigma_{noise}^{\ 2}}}$$

and *Amp* is estimated from the event magnitude, σ_{amp} is the model error, and σ_{noise} is the noise standard deviation. The *snr-thresh* is the user-supplied signal-to-noise ratio threshold. For a more complete discussion of the detection probability computation see [IDC5.2.1].

Control

The Station Detection Probability functions *get_det_prob* and *get_regional_ prob* are called by the function *set_beta_station_attributes,* a sub-function of *process_select_mode* and *process_ipc_mode* (processes 3 and 4 in Figure 8). Normal or abnormal termination of the functions occurs when the function status is returned to *set_beta_station_attributes.*

Interfaces

The Station Detection Probability routines interface with the *libprob* and *libmagnitude* global library functions, respectively, to obtain probability and magnitude information. The *libprob* function *prob_of_detect* takes station distances, thresholds, noise levels, and attenuation information from the attenuation file specified by user parameter, and returns the array of station probabilities. The

function *prob_func* takes the value of *X* (described in section "Input/Processing/Output" on page 38), which is computed for a specific station using the *libmagnitude station_magnitude* function to estimate the station and returns the probability.

Error States

Failure during the regional probability computation is non-fatal, because the general probability value will be used as a replacement. Failure during regional processing can occur if problems arise during the regional magnitude computation (*libmagnitude* function *station_magnitude*) or if the σ_{amp} and σ_{noise} values are exceedingly small. Failure during general probability computation may occur if memory is insufficient, if the event has no valid magnitude, or if an error occurs in the *libprob* function *prob_of_detect.*

Processing failures cause an error message to be printed and an error status to be returned to the calling function, which in turn prints an error message and passes an error status to its parent. When the processing mode is `ipc-assess`, an error in the detection probability computations causes further processing on the origin to be skipped, and *WaveExpert* waits for the next IPC message. When the processing mode is `auto`, the error is propagated to the *main* routine and causes *WaveExpert* to exit with an error status.

Default Station Ranking

The Default Station Ranking process provides an assessment of how much the inclusion of a station is likely to improve the constraint on the existing event location using the default ranking algorithm.

Input/Processing/Output

Input to Default Station Ranking is provided by the `BSta` array and `Alpha_Event` input arguments to the function *jacknife* (called from *set_beta_station_ attributes,* a subfunction of *process_orid_select*, process 7 in Figure 8) and by runtime parame-

ters. A key value used in *jacknife* (which uses a type of jackknife algorithm) is the station detection probability stored in the `det_prob` member of a station's `Bsta` structure.

Auxiliary stations are sorted by detection probability and are tested one-by-one for inclusion in the working set of event associations referred to as the "core network." The core network is comprised of existing associated stations and auxiliary stations that meet the inclusion criteria. If the inclusion of a station will significantly improve the event location, then the station is added to the core network, and the next station is processed. Processing terminates after all stations having detection probabilities above a specified threshold are checked. The detection probability and event location improvement thresholds used to specify the inclusion criteria in Default Station Ranking are specified by user parameters.

Output from Default Station Ranking is provided by setting the station importance value and location improvement values to the *jacknife* and `err_vol_change` `Bsta` structure members respectively for each station passing the inclusion criteria.

Control

The Default Station Ranking function *jacknife* is called from the function *set_beta_station_attributes.* The default ranking and user-specific rankings are done in the function *rank_beta_stations.* These functions are all subfunctions of the function *process_orid_select* (process 7 in Figure 8). The Default Station Ranking functions terminate by returning the processing status to their calling functions.

Interfaces

Data are exchanged within the Default Station Ranking functions by normal assignment statements and calls to subfunctions using the `Bsta` and `Alpha_Event` structures (M1 and M2 in Figures 7 and 8, and described in Tables 4 and 5).

Error States

Failure during Default Station Ranking will occur if the *libloc* functions have event data inconsistencies, memory allocation problems, or problems in the *libloc* functions. Processing failures cause an error status to be returned to the calling function, and an appropriate error message is output to the log file. When the processing mode is `ipc-assess`, an error in the default station ranking computations causes further processing on the origin to be skipped, and *WaveExpert* waits for the next IPC message. When the processing mode is `auto`, the error is propagated to the *main* routine and causes *WaveExpert* to exit with an error status.

Rule-based Station Ranking

The Rule-based Station Ranking process provides a flexible mechanism to the user to specify the way in which the station ranks and hence the final request set are assigned. The *heart* of this process is a *CLIPS* rule-based system, which has been incorporated into *WaveExpert*. Rules can make inference based on station-specific data to override or supplement the default ranking. Programming methods used in rule-based systems are beyond the scope of this document. See [Ril98] for information on programming in *CLIPS*.

Input/Processing/Output

Inputs to the Rule-based Station Ranking process are provided by information contained in the station `Bsta` structures and rules contained in the files indicated by the runtime parameters *init_rule_file* and *rule_file*. The `Bsta` structure member values are used to create *CLIPS facts,* and any defined rules will use the facts (enumerated in "Rule-based Station Ranking" on page 24) for the rule-based processing. After the rule processing has finished, the station ranks are output by setting the value of the `Bsta` *rank* member, superseding the value previously assigned during Default Station Ranking.

Control

All rule-based station ranking is done in the *rank_beta_stations* function, which is called by *process_orid_select* (see process 7, Figure 8), the main top-level function for normal IDC processing. Rule-based processing is only done if the *rule_file* parameter is set; otherwise, the values computed during Default Station Ranking are used. Control is returned to *process_orid_select* through a status return.

Interfaces

Station attributes stored in the `Bsta` structure are assigned to *CLIPS* facts by the function *assign_station_facts*. Rank values resulting from rule processing are obtained by the function *find_ranks*. The rank value interface functions (functions callable from the *CLIPS* rules) are defined in the file `station_rank.c`. Available facts set by *assign_station_facts* and a sample rule are shown in "Rule-based Station Ranking" on page 24.

Error States

The most likely cause of errors during Rule-based Station Ranking will be missing or incorrectly named rule files and syntactic errors in the user rules. These problems will cause an error status to be returned to the calling function and an appropriate error message to be printed to the log file. When the processing mode is `ipc-assess`, an error in the rule-based computations causes further processing on the origin to be skipped, and *WaveExpert* waits for the next IPC message. When the processing mode is `auto`, the error is propagated to the *main* routine and causes *WaveExpert* to exit with an error status.

Station Interval Computation

The Station Interval Computation process (processes 8 and 9 in Figure 7 and processes 9 and 10 in Figure 8) computes the time interval for each station data request. These intervals are refined by excluding time segments already existing in the **request** and **wfdisc** tables.

Input/Processing/Output

The first step in Station Interval Computation process is to predict the time interval in which the phases of interest will be contained. This is done in the function *get_interval_for_origin_station,* which takes the origin and station `Bsta` structures as input arguments. A set of phases are specified by a runtime parameter, and only phases valid for the station-to-event distance are included. Phase travel-times are computed in the *trv_time_w_ellip_elev libloc* function using travel time tables referenced by the velocity model specification file parameter *vmsf*. The time interval is computed using the fastest and slowest phase travel-times plus lead and lag times specified by parameters. The resulting interval is returned by setting the `proc_intervals` member of the station's `Bsta` structure.

Next, the function *set_final_request_intervals* modifies the original interval by removing portions that already exist either in the **request** or **wfdisc** tables. Arguments to *set_final_request_intervals* include the array of station `Bsta` structures, the database connection object, and the **request** and **wfdisc** table names. The subfunctions *get_wfdisc_intervals* and *get_interval_intervals* obtain respectively the existing **wfdisc** and **request** intervals.

The modified intervals are output by setting the `proc_intervals` member of the station's `Bsta` structure.

Control

Station Interval Computation is initiated by a function call to *set_final_request_intervals* (processes 7 and 9 in Figures 7 and 8), by the function *process_orid_select* when a station is selected (Figure 7), or by *process_all_mode* when only intervals are computed (Figure 8).

Interfaces

Station Interval Computation obtains computed intervals from its function parameters and existing intervals from the output database. The **wfdisc** and **request** tables are accessed through *libgdi* functions.

Error States

The most likely causes of errors in the Station Interval Computation process are memory allocation problems and database access problems. Both conditions are handled by printing appropriate messages to the log file and returning an error status to the calling routine. When the processing mode is `ipc-request`, an error in the computations causes further processing on the origin to be skipped, and *WaveExpert* waits for the next IPC message. When the processing mode is `auto` or `interval`, the error is propagated to the *main* routine and causes *WaveExpert* to exit with an error status.

Interval Output

The Interval Output (processes 11 and 10 in Figures 7 and 8) inserts the intervals computed in the *Station Interval Computation* process into the **request** tables via *libgdi* functions.

Input/Processing/Output

Input to this process is provided through function arguments and user parameters. The function *write_request_intervals* takes arguments providing `Bsta` structures, which contain the station-intervals, **request** table name, and origin ID and event ID values. Parameters provide static **request** table attributes (such as class and state) and a flag to write requests for all array elements, or just the station.

The first step in Interval Output processing is to determine the stations, station elements, and components to be requested. A loop is made over each station and, if indicated by the parameter `expand_array`, all affiliated array elements are obtained from the **affiliation** table. Next, the components are determined from general or station-specific user parameters and **sitechan** information. Finally, a **request** record is written for each interval, element, and component for the station, and then the next station is processed.

Control

Interval Output is initiated by a function call to *write_request_intervals* by the functions *process_orid_select* and *process_all_mode.* Errors in *write_request_intervals* or its subfunctions cause termination with an error status, otherwise normal status is returned on termination.

Interfaces

The Interval Output processes deal mainly with the station elements to be requested. These data are obtained from the database and user parameters. The database **affiliation** table provides the station elements, and the **sitechan** table and user parameters provide the component information. All interfaces to the database are managed by functions in *libgdi*.

Error States

The most likely causes of errors in the Interval Output process are database access problems, which are handled by printing appropriate messages to the log file and returning an error status to the calling routine.

DATA STRUCTURES

WaveExpert uses three primary internal data stores, objects M1, M2, and M3, as shown in Figure 7. The `Alpha_Event` (M1) structure stores all information pertaining to the event being processed. The `Bsta` (M2) structure stores all information pertaining to an auxiliary station for convenient access by processing units. The `Origin_Info` (M3) structure stores origin attributes.

Key *WaveExpert* data structures are shown in Tables 4 through 6.

TABLE 4: BSTA DATA STRUCTURE–AUXILIARY STATION INFORMATION

Name	Type	Description
sta	char[7]	station name
ttime	double	travel time
lat	float	station latitude
lon	float	station longitude
noise	float	regional noise level from **siteaux** table
noise_sd	float	regional noise level standard deviation
rely	float	regional reliability from **siteaux** table
tele_noise	float	teleseismic noise level from **siteaux** table
tele_noise_sd	float	teleseismic noise level standard deviation
tele_rely	float	teleseismic reliability from **siteaux** table
azimuth	float	event to station azimuth
distance	float	event to station distance in degrees
import	float	data importance while including all eligible auxiliary stations
solo_import	float	data importance while including only this station with existing associations
det_prob	float	detection probability
gap_redux	float	gap reduction by inclusion of station
jacknife	float	importance value from *jackknife* algorithm
err_vol_change	float	change in error ellipsoid volume due to station inclusion
rank	float	final station rank value
site_record	Site[1]	site record for this station

TABLE 4: BSTA DATA STRUCTURE–AUXILIARY STATION INFORMATION (CONTINUED)

Name	Type	Description
sitechan_records	*Sitechan[2]	**sitechan** records for this station
nsitechan_records	int	number of **sitechan** records
proc_intervals	Proc_Ints[3]	data intervals computed for station

1. Site is a typedef structure defined in the global include file site_Astructs.h and contains members corresponding to **site** table attributes.
2. Sitechan is a typedef structure defined in the global include file sitechan_Astructs.h and contains members corresponding to **sitechan** table attributes. The preceding asterisk (*) indicates that this member is a reference, in this case a reference to an array of nsitechan_records **sitechan** records.
3. Proc_Ints is a typedef structure local to *WaveExpert* that provides convenient storage for time intervals.

TABLE 5: ALPHA_EVENT DATA STRUCTURE–ORIGIN ASSOCIATED INFORMATION

Name	Type	Description
origin_record	*Origin[1]	**origin** record
origerr_record	*Origerr[2]	**origerr** record
assoc_records	*Assoc[3]	**assoc** records
nassoc	int	number of **assoc** records
arrival_records	*Arrival[4]	**arrival** records
narrival	int	number of **arrival** records
site_records	*Site	**site** records
nsite	nsite	number of **site** records

1. Origin is a typedef structure defined in the global include file origin_Astructs.h and contains members corresponding to **origin** table attributes. The preceding asterisk (*) indicates that this member is a reference.

2. `Origerr` is a typedef structure defined in the global include file `origerr_Astructs.h` and contains members corresponding to **origerr** table attributes. The preceding asterisk `(*)` indicates that this member is a reference.
3. `Assoc` is a typedef structure defined in the global include file `assoc_Astructs.h` and contains members corresponding to **assoc** table attributes. The preceding asterisk `(*)` indicates that this member is a reference, in this case a reference to an array of `nassoc` **assoc** records.
4. `Arrival` is a typedef structure defined in the global include file `arrival_Astructs.h` and contains members corresponding to **arrival** table attributes. The preceding asterisk `(*)` indicates that this member is a reference, in this case a reference to an array of `narrival` **arrival** records.

TABLE 6: ORIGIN_INFO DATA STRUCTURE–ORIGIN INFORMATION

Name	Type	Description
`orid`	long	origin ID
`evid`	long	event ID
`lat`	float	event latitude
`lon`	float	event longitude
`depth`	float	event depth
`time`	int	origin time
`jdate`	long	origin julian date
`mb`	double	body-wave magnitude
`ms`	double	surface-wave magnitude
`ml`	double	local magnitude

DATABASE DESCRIPTION

As described in previous sections, *WaveExpert* relies on the database for input data and as a repository for its output. All database interaction is through *libgdi* function calls. This section describes database design, the database entity-relationship model, and database schema.

Database Design

WaveExpert uses IDC database tables for processing. See *IDC Database Schema* [IDC5.1.1Rev2] for a description of all tables and for entity-relationship diagrams. The entity-relationship model relevant to *WaveExpert* is indicated in Figure 9.

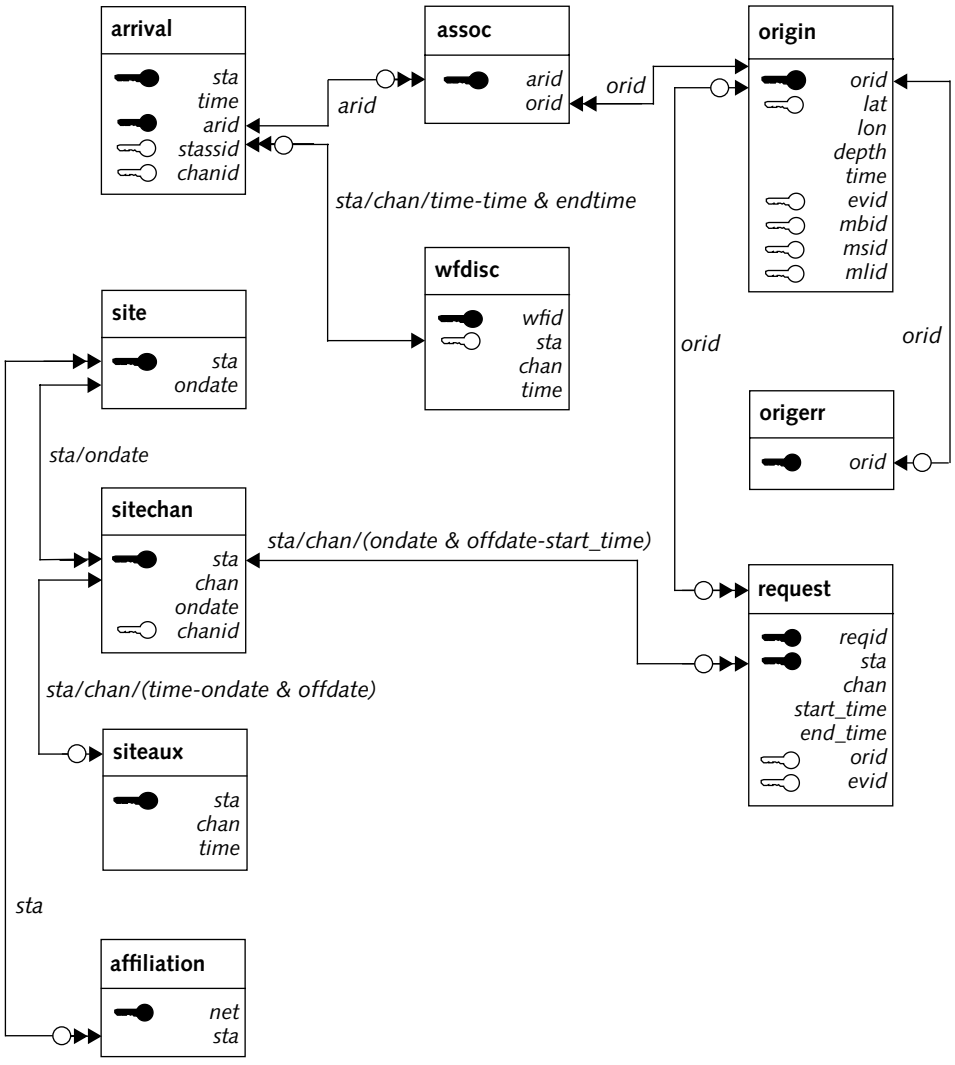


FIGURE 9. WAVEEXPERT TABLE RELATIONSHIPS

Database Schema

Table 7 shows the usage of database tables by WaveExpert. For each table used, the third column shows the purpose for reading or writing each attribute.

TABLE 7: DATABASE USAGE BY WAVEEXPERT

Table	Action	Usage of Attributes
origin	reads	• *orid* and *evid* for record identification • *lat*, *lon*, and *depth* for computing travel times and station ranking using *libloc* functions • *time* for computing intervals • *jdate* for constraint in querying **site** and **sitechan** • *mb*, *ms*, and *ml* for computing detection probability
origerr	reads	• *orid* for record identification • *sxx*, *syy*, *szz*, *stt*, *sxy*, *sxz*, *syz*, *stx*, *sty*, *stz*, *sdobs*, *smajax*, *siminax*, *strike*, *sdepth*, *stime*, and *conf* for computing station ranking using *libloc* functions
assoc	reads	• *arid* and *orid* for record identification • *sta*, *phase*, *belief*, *delta*, *seaz*, *esaz*, *timers*, *timedef*, *azres*, *azdef*, *slores*, *slodef*, *emares*, *wgt*, and *vmodel* for computing station ranking using *libloc* functions
arrival	reads	• *arid* for record identification • *sta*, *time*, *iphase*, *deltim*, *azimuth*, *delaz*, *slow*, and *delslo* for computing station ranking using *libloc* functions
site	reads	• *sta*, *ondate*, and *offdate* for record identification • *lat*, *lon*, *elev* for computing station ranking
sitechan	reads	• *sta*, *ondate*, and *offdate* for record identification
siteaux	reads	• *sta*, *chan*, and *staper* for record identification • *nois*, *noissd*, and *rely* for detection probability
affiliation	reads	• *net* for record identification • *sta* for identifying auxiliary network, or station elements for creating station intervals

TABLE 7: DATABASE USAGE BY WAVEEXPERT (CONTINUED)

Table	Action	Usage of Attributes
wfdisc	reads	• *sta* and *chan* for record identification • *time* and *endtime* to identify existing data intervals to tailor creation of station intervals
request	reads	• all attributes to obtain existing station intervals, which are then omitted from the computed request interval to avoid unnecessary data requests
	writes	• all attributes to define the station-channel-time intervals of data to be retrieved from auxiliary stations

References

The following sources supplement or are referenced in this document:

[DOD94a] Department of Defense, "Software Design Description," *Military Standard Software Development and Documentation,* MIL-STD-498, 1994.

[DOD94b] Department of Defense, "Software Requirements Specification," *Military Standard Software Development and Documentation,* MIL-STD-498, 1994.

[Fox94] Fox, W., *Beta Request System*, CCB-PRO-94/21, 1994.

[Gan79] Gane, C., and Sarson, T., *Structured Systems Analysis: Tools and Techniques*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1979.

[IDC3.4.1Rev2] Science Applications International Corporation, Veridian Pacific-Sierra Research, *Formats and Protocols for Messages, Revision 2*, SAIC-00/3005, PSR-00/TN2829, 2000.

[IDC5.2.1] Science Applications International Corporation, *IDC Processing of Seismic, Hydroacoustic, and Infrasonic Data*, SAIC-99/3023, 1999.

[IDC5.1.1Rev2] Science Applications International Corporation, Veridian Pacific-Sierra Research, *Database Schema, Revision 2*, SAIC-00/3057, PSR-00/TN2830, 2000.

[Ril98] Riley, G., Culbert, C., and Donnell, B., *CLIPS Reference Manual,* Volume I, GHG Internet Services, 1998. (online) `http://www.ghgcorp.com/clips/CLIPS.html`

Glossary

Symbols

3-C

Three-component.

A

amp

Amplitude.

amplitude

Zero-to-peak height of a waveform in nanometers.

architecture

Organizational structure of a system or component.

architectural design

Collection of hardware and software components and their interfaces to establish the framework for the development of a computer system.

array

Collection of sensors distributed over a finite area (usually in a cross or concentric pattern) and referred to as a single station.

arrival

Signal that has been associated to an event. First, the Global Association (GA) software associates the signal to an event. Later during interactive processing, many arrivals are confirmed and improved by visual inspection.

ARS

Analyst Review Station. This application provides tools for a human analyst to refine and improve the event bulletin by interactive analysis.

ASCII

American Standard Code for Information Interchange. Standard, unformatted 256-character set of letters and numbers.

associate

Assign an arrival to an event.

associated phase

Phase that is associated with an S/H/I event.

attribute

(1) A database column. (2) Characteristic of an item; specifically, a quantitative measure of a S/H/I arrival such as azimuth, slowness, period, and amplitude.

azimuth

Direction, in degrees clockwise with respect to North, from a station to an event.

B

back azimuth

Direction, in degrees, from an event to the station.

background noise

Natural movements of the earth, oceans, and atmosphere as seen on S/H/I sensors (usually measured in data preceding signals).

bulletin

Chronological listing of event origins spanning an interval of time. Often, the specification of each origin or event is accompanied by the event's arrivals and sometimes with the event's waveforms.

C

chan.

Channel.

change request

The recognized procedure for reporting software problems or requesting upgrades.

channel

Component of motion or distinct stream of data.

CLIPS

"C" Language Integrated Production System. An Expert System language that was created by NASA's Software Technology Branch.

CMR

Center for Monitoring Research.

command

Expression that can be input to a computer system to initiate an action or affect the execution of a computer program.

component

(1) One dimension of a three-dimensional signal; (2) The vertically or horizontally oriented (north or east) sensor of a station used to measure the dimension; (3) One of the parts of a system; also referred to as a module or unit.

Comprehensive Nuclear-Test-Ban Treaty Organization

Treaty User group that consists of the Conference of States Parties (CSP), the Executive Council, and the Technical Secretariat.

Computer Software Component

Functionally or logically distinct part of a computer software configuration item, typically an aggregate of two or more software units.

Computer Software Configuration Item

Aggregation of software that is designated for configuration management and treated as a single entity in the configuration management process.

Conference of States Parties

Principal body of the CTBTO consisting of one representative from each State Party accompanied by alternate representatives and advisers. The CSP is responsible for implementing, executing, and verifying compliance with the Treaty.

configuration

Arrangement of computer system or component as defined by the number, nature, and interconnection of its parts.

configuration item

Aggregation of hardware, software, or both treated as a single entity in the configuration management process.

configuration management

Directing and surveying the functional and physical characteristics of a configuration item, controlling changes to those characteristics, and recording and reporting changes and implementation, and verifying compliance with requirements.

control flow

Sequence in which operations are performed during the execution of a computer program.

COTS

Commercial-Off-the-Shelf; terminology that designates products such as hardware or software that can be acquired from existing inventory and used without modification.

CPU

Central Processing Unit.

CR

See Change Request.

CSC

Computer Software Component.

CSCI

Computer Software Configuration Item.

CTBTO

Comprehensive Nuclear-Test-Ban Treaty Organization; Treaty User group that consists of the Conference of States Parties (CSP), the Executive Council, and the Technical Secretariat.

D

data flow

Sequence in which data are transferred, used, and transformed during the execution of a computer program.

data type

Kind of data in a data message. S/H/I data types include: ARRIVAL, BULLETIN, CHANNEL, CHAN_STATUS, COMMENT, COMM_STATUS, EVENT, EXEC-

SUM, NETWORK ORIGIN, OUTAGE, RESPONSE STATION, STA_STATUS, and WAVEFORM

DBMS

Database Management System.

detailed design

Refined and expanded version of the preliminary design of a system or component. This design is complete enough to be implemented.

detection

Probable signal that has been automatically detected by the *Detection and Feature Extraction* (*DFX*) software.

DFX

Detection and Feature Extraction.

E

entity-relationship (E-R) diagram

Diagram that depicts a set of entities and the logical relationships among them.

event

Unique source of seismic, hydroacoustic, or infrasonic wave energy that is limited in both time and space.

execute

Carry out an instruction, process, or computer program.

exit status

Value returned at the completion of a UNIX command.

F

failure

Inability of a system or component to perform its required functions within specified performance requirements.

filesystem

Named structure containing files in subdirectories. For example, UNIX can support many filesystems; each has a unique name and can be attached (or mounted) anywhere in the existing file structure.

FTP

File Transfer Protocol; protocol for transferring files between computers.

G

GA

Global Association application. GA associates S/H/I phases to events.

GB

Gigabyte. A measure of computer memory or disk space that is equal to 1,024 megabytes.

GMT

Greenwich Mean Time.

GSETT

Group of Scientific Experts Technical Test.

I

ID

Identification; identifier.

IDC

International Data Centre.

IMS

International Monitoring System.

IPC

Interprocess communication. The messaging system by which applications communicate with each other through *libipc* common library functions. See *tuxshell*.

J

Julian date

Increasing count of the number of days since an arbitrary starting date.

L

LAN

Local Area Network.

libgdi

Library containing functions for RDBMS access.

local

(1) (distance) Source to seismometer separations of a few degrees or less. (2) (event) Recorded at distances where the first P and S waves from shallow events have traveled along direct paths within the crust.

M

Map

Application for displaying S/H/I events, stations, and other information on geographical maps.

MB

Megabyte. 1,024 kilobytes.

monitoring system

See IMS.

N

noise

Incoherent natural or artificial perturbations of the waveform trace caused by ice, animals migrations, cultural activity, equipment malfunctions or interruption of satellite communication, or ambient background movements.

O

ORACLE

Vendor of PIDC and IDC database management system.

orid

Origin Identifier.

origin

Hypothesized time and location of a seismic, hydroacoustic, or infrasonic event. Any event may have many origins. Characteristics such as magnitudes and error estimates may be associated with an origin.

P

par

See parameter.

parameter

User-specified token that controls some aspect of an application (for example, database name, threshold value). Most parameters are specified using [*token* = *value*] strings, for example, `dbname=mydata/base@oracle`.

parameter (par) file

ASCII file containing values for parameters of a program. Par files are used to replace command line arguments. The files are formatted as a list of [*token* = *value*] strings.

period

Average duration of one cycle of a phase, in seconds per cycle.

phase

Arrival that is identified based on its path through the earth.

phase name

Name assigned to a seismic, hydroacoustic or infrasonic arrival associated with a travel path.

PIDC

Prototype International Data Centre.

pipeline

Flow of data at the IDC from the receipt of communications to the final automated processed data before analyst review.

primary seismic

IMS seismic station(s) or data that is (are) part of the detection network.

probability of detection

Probability that a signal from a given event will have an amplitude significantly greater than the background noise level.

R

RAM

Random Access Memory.

RDBMS

Relational Database Management System.

regional

(1) (distance) Source to seismometer separations between a few degrees and 20 degrees. (2) (event) Recorded at distances where the first P and S waves from shallow events have traveled along paths through the uppermost mantle.

run

(1) Single, usually continuous, execution of a computer program. (2) To execute a computer program.

S

schema

Database structure description.

S/H/I

Seismic, hydroacoustic, and infrasonic.

snr

Signal-to-noise ratio.

software unit

Discrete set of software statements that implements a function; usually a sub-component of a CSC.

Solaris

Name of the operating system used on Sun Microsystems hardware.

StaPro

Station Processing application for S/H/I data.

States Parties

Treaty user group who will operate their own or cooperative facilities, which may be NDCs.

station

Site where a monitoring instrument is installed. Stations can either be single sites (for example, BGCA) or arrays (for example, ASAR).

subsystem

Secondary or subordinate system within the larger system.

T

teleseismic

(1) (distance) Source to seismometer separations of 20 degrees or more. (2) (event) Recorded at distances where the first P and S waves from shallow events have traveled paths through the mantle/core.

theoretical arrival

Point where an arrival is expected to appear on a waveform, based on an event's location and depth.

tuxshell

Process used to execute applications lacking interprocess communications capabilities through Tuxedo. See IPC.

typdef

'C' programming language method of declaring new data types, based either on intrinsic C datatypes or previously defined user types.

U

UNIX

Trade name of the operating system used by the Sun workstations.

V

VMSF

Velocity model specification file (for setting travel time models in S/H/I event location).

W

wfdisc

Waveform description record or table.

WorkFlow

Software that displays the progress of automated processing systems.

Index

I

J

L

M

N

O

P

R

S

T

U

V

W